

Redis

[Redis简介](#)

[Redis下载与安装](#)

[Anthon-Redis-Desktop-Manager安装](#)

[Redis的数据类型](#)

[Redis中常用命令](#)

[字符串操作命令](#)

[哈希操作命令](#)

[列表操作命令](#)

[集合操作命令](#)

[有序集合操作命令](#)

[通用命令](#)

[在Java中操作Redis](#)

[Redis的java客户端](#)

[Sping Data Redis使用方式](#)

Redis简介

Redis是一个基于内存的key-value（键值对）结构数据库

key	value
id	101
name	小智
city	北京
	↙

- 基于内存存储，读写性能高
- 访问量大的适合存储热点数据（热点商品、咨询、新闻）
- 企业应用广泛

官网：<http://redis.io>

中文网站：<http://www.redis.net.cn/>

Redis下载与安装

window下载地址：<https://github.com/microsoftarchive/redis/releases>

	EventLog.dll	2016/7/1 16:27	应用程序扩展	1 KB
	Redis on Windows Release Notes.docx	2016/7/1 16:07	DOCX 文档	13 KB
	Redis on Windows.docx	2016/7/1 16:07	DOCX 文档	17 KB
	redis.windows.conf redis配置文件	2016/7/1 16:07	CONF 文件	48 KB
	redis.windows-service.conf	2016/7/1 16:07	CONF 文件	48 KB
	redis-benchmark.exe	2016/7/1 16:28	应用程序	400 KB
	redis-benchmark.pdb	2016/7/1 16:28	PDB 文件	4,268 KB
	redis-check-aof.exe	2016/7/1 16:28	应用程序	251 KB
	redis-check-aof.pdb	2016/7/1 16:28	PDB 文件	3,436 KB
	redis-cli.exe redis客户端	2016/7/1 16:28	应用程序	488 KB
	redis-cli.pdb	2016/7/1 16:28	PDB 文件	4,420 KB
	redis-server.exe redis服务端	2016/7/1 16:28	应用程序	1,628 KB
	redis-server.pdb	2016/7/1 16:28	PDB 文件	6,916 KB
	Windows Service Documentation.docx	2016/7/1 9:17	DOCX 文档	14 KB

服务端：

```
Microsoft Windows [版本 10.0_22631_4037]
(c) Microsoft Corporation。保留所有权利。

C:\Users\zengzicong\Desktop\新建文件夹\资料\day05\Redis-x64-3.2.100>redis-server.exe redis.windows.conf

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 23920

http://redis.io

[23920] 21 Aug 14:45:15.783 # Server started, Redis version 3.2.100
[23920] 21 Aug 14:45:15.783 * The server is now ready to accept connections on port 6379
```

ctrl + c : 结束进程

Microsoft Windows [版本 10.0.22631.4037]
(c) Microsoft Corporation。保留所有权利。

C:\Users\zengzicong\Desktop\新建文件夹\资料\day05\Redis-x64-3.2.100>redis-server.exe redis.windows.conf

Redis 3.2.100 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 23920

http://redis.io

[23920] 21 Aug 14:45:15.783 # Server started, Redis version 3.2.100
[23920] 21 Aug 14:45:15.783 * The server is now ready to accept connections on port 6379
[23920] 21 Aug 14:45:54.778 # User requested shutdown...
[23920] 21 Aug 14:45:54.778 * Saving the final RDB snapshot before exiting.
[23920] 21 Aug 14:45:54.781 * DB saved on disk
[23920] 21 Aug 14:45:54.781 # Redis is now ready to exit, bye bye...

C:\Users\zengzicong\Desktop\新建文件夹\资料\day05\Redis-x64-3.2.100>

客户端：exit退出

D:\software\Redis-x64-3.2.100>redis-cli.exe -h localhost -p 6379 客户端
localhost:6379> keys *
(error) NOAUTH Authentication required.
localhost:6379> exit

D:\software\Redis-x64-3.2.100>redis-cli.exe -h localhost -p 6379 -a 123456
localhost:6379> keys *
(empty list or set) 地址 任务号 密码
localhost:6379> -

Anthon-Redis-Desktop-Manager安装

Another Redis Desktop Manager

localhost

+ 新增Key

Enter 键进行搜索

加载更多

第一步

Redis版本: 3.2.100
OS: Windows
进程ID: 13252

已用内存: 694.89K
内存占用峰值: 694.89K
Lua占用内存: 37K

客户端连接数: 2
历史连接数: 2
历史命令数: 5

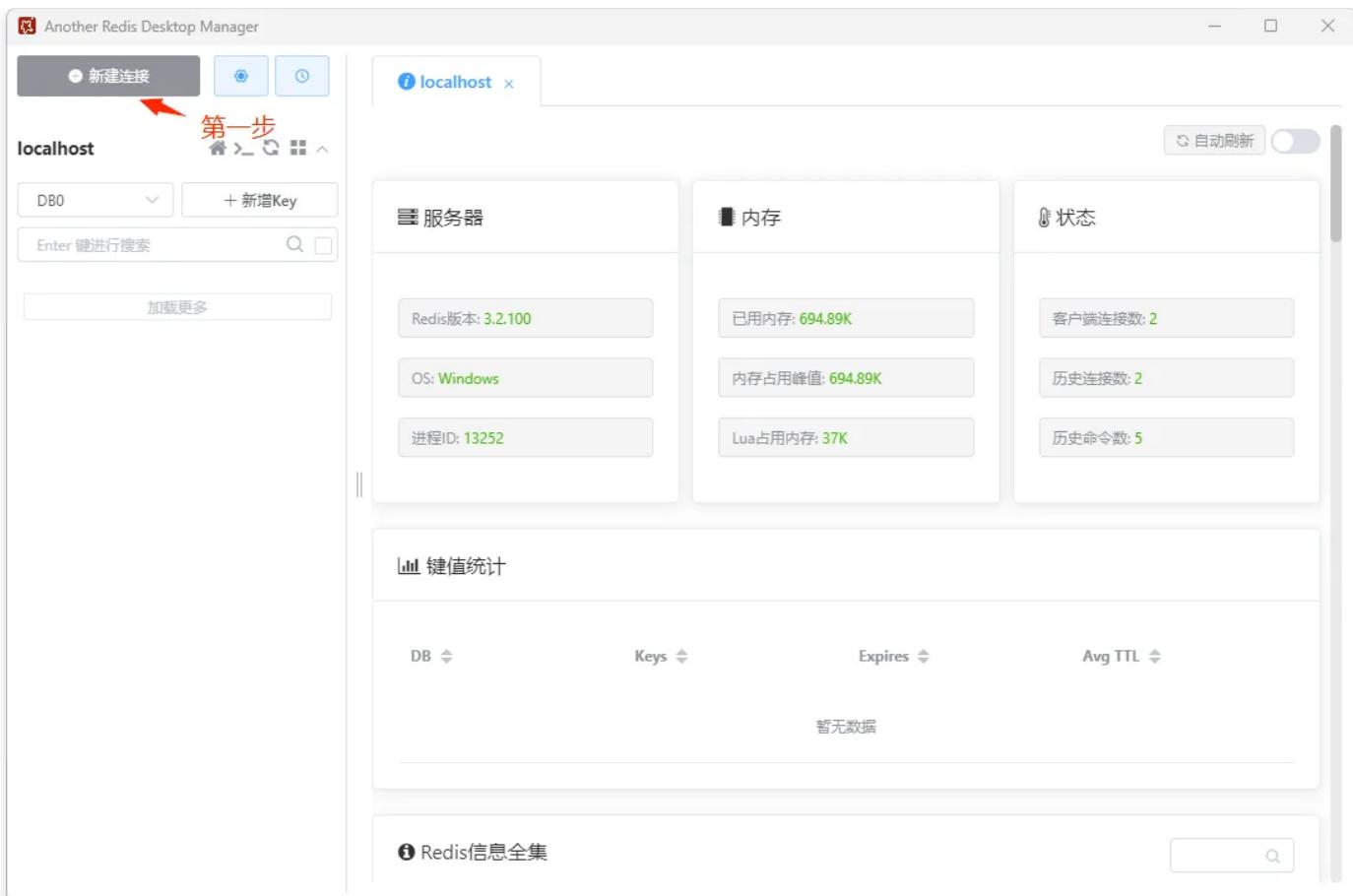
键值统计

DB Keys Expires Avg TTL

暂无数据

Redis信息全集

自动刷新



localhost

+ 新增Key

Enter 键进行搜索

新建连接

* 地址: 127.0.0.1 * 端口: 6379

密码

连接名称

用户名

ACL in Redis >= 6.0

分隔符:

SSH SSL Sentinel Cluster Readonly

只需填这几个就行

取消 确定

Redis信息全集



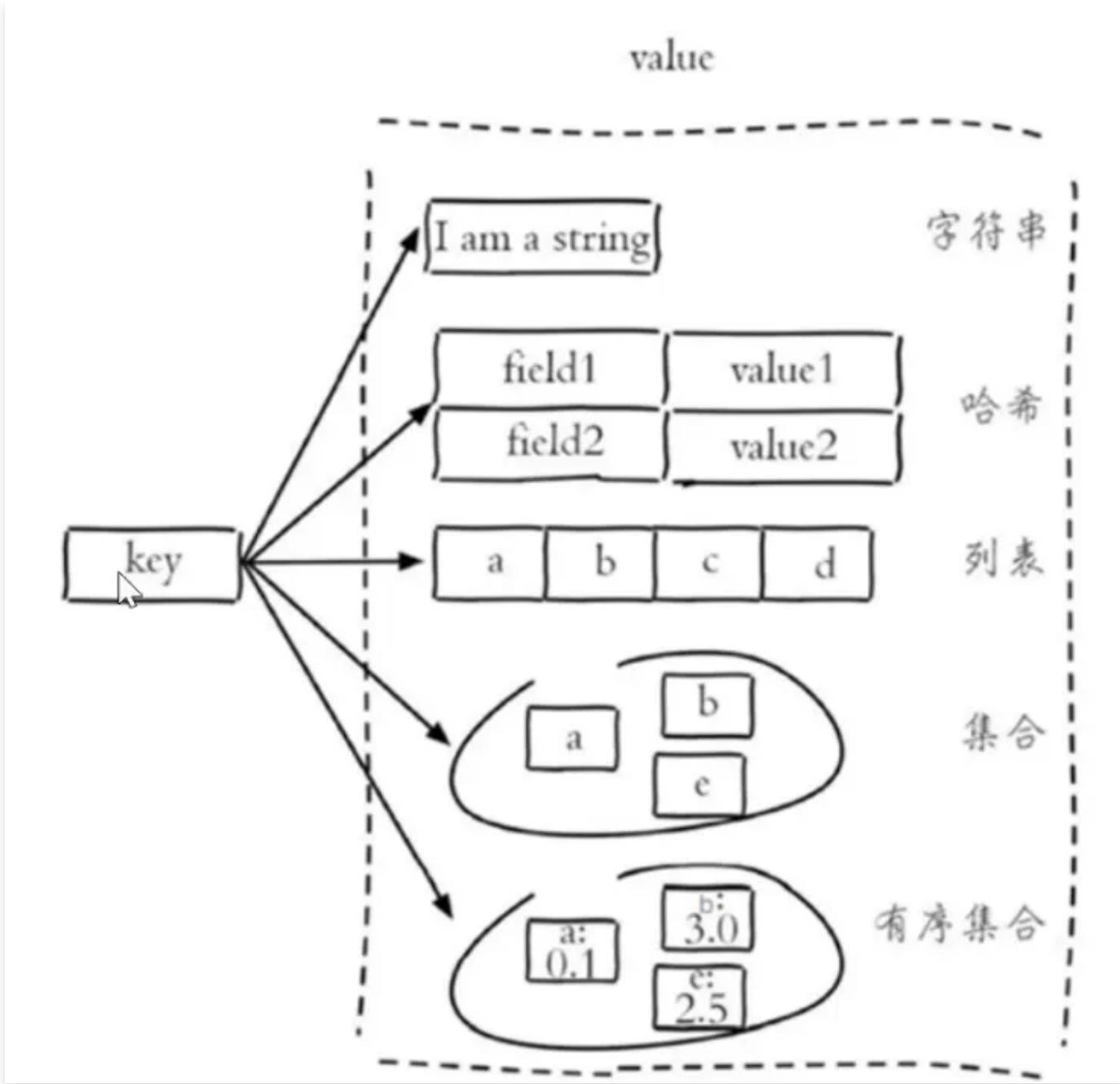
Redis的数据类型

1. 5种常用数据类型

Redis存储的是Key-value结构的数据，其中key是字符串类型，value有5种常用的数据类型：

- 字符串 String
- 哈希 hash
- 列表 list
- 集合 set
- 有序集合 sorted set / zset

2. 各种数据类型的特点



字符串 (string) : 普通字符串, redis中最简单的数据类型

哈希 (hash) : 也叫散列, 类似于java中的HashMap结构

列表 (list) : 按照插入顺序排序, 可以有重复元素, 类似于java中的LinkedList

集合 (set) : 无序集合, 没有重复元素, 类似于java中的HashSet

有序集合 (sorted set / zset) : 集合中每个元素关联一个分数 (score) , 根据分数升序排序, 没有重复元素

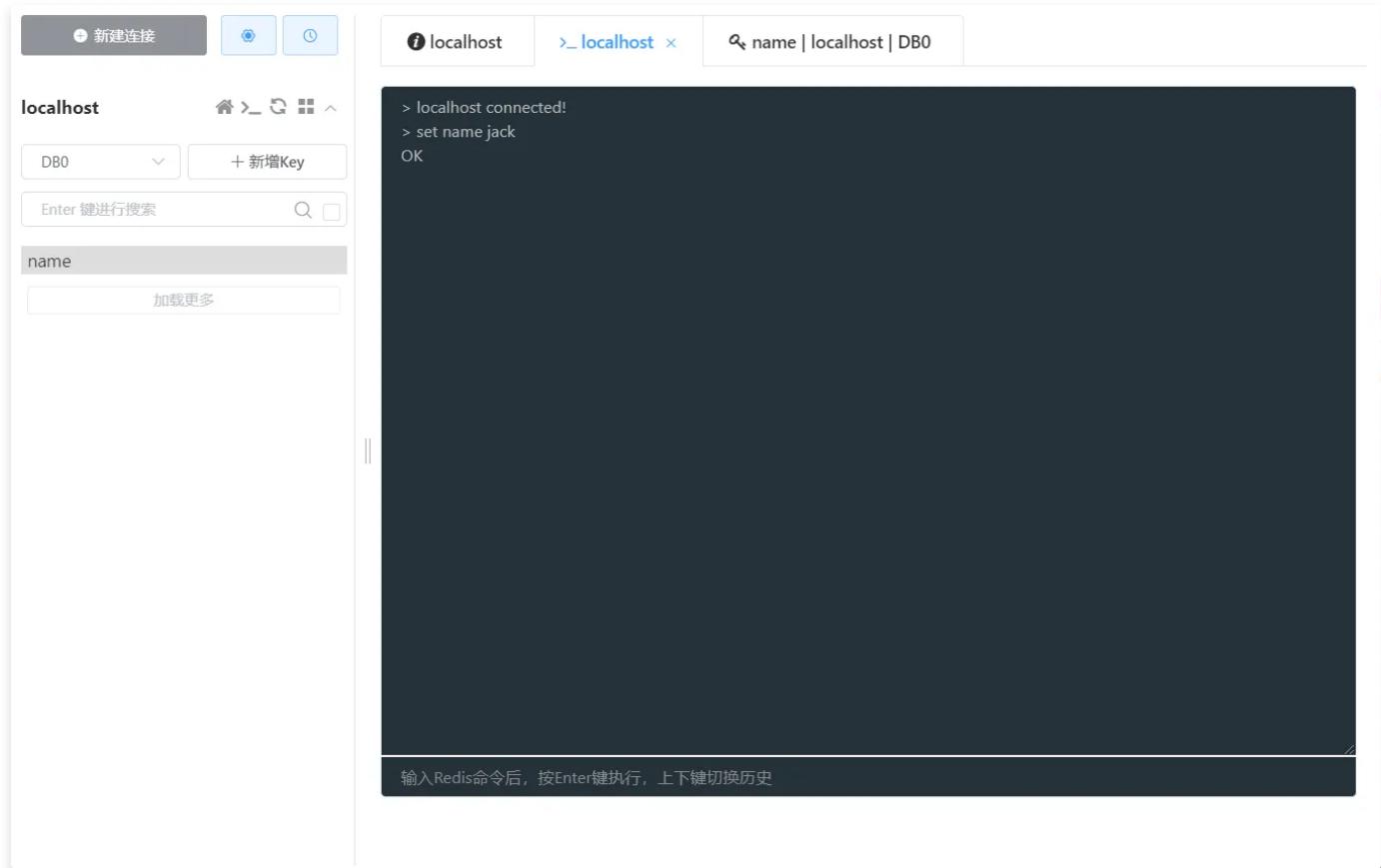
Redis中常用命令

- 字符串操作命令
- 哈希操作命令
- 列表操作命令
- 集合操作命令
- 有序集合操作命令
- 通用操作命令

字符串操作命令

Redis字符串类型常用命令：

- | | |
|------------------------------------|---|
| • SET key value | 设置指定key的值 |
| • GET key | 获取指定key的值 |
| • SETEX key seconds value
会自动删除 | 设置指定key的值，并将key的过期时间设置为seconds秒，
会自动删除 |
| • SETNX key value | 只有在key不存在时设置key的值 |



The screenshot shows the Redis UI interface. On the left, the sidebar displays the connection information: 'localhost' (selected), 'DB0' (selected), and '+ 新增Key'. The search bar contains 'Enter 键进行搜索'. Below it, a table lists a single row with the key 'name' and the value 'jack'. A '加载更多' button is at the bottom of the table. On the right, the main panel shows the key-value pair 'name' with value 'jack'. The key type is 'String' and the size is '4B'. The TTL is set to '-1'. Below the table is a large input area containing the value 'jack'. At the bottom of this area is a blue '保存' (Save) button.

The screenshot shows the Redis UI interface. On the left, the sidebar displays the connection information: 'localhost' (selected), 'DB0' (selected), and '+ 新增Key'. The search bar contains 'Enter 键进行搜索'. Below it, a table lists a single row with the key 'code' and the value 'name'. A '加载更多' button is at the bottom of the table. On the right, the main panel shows a command history window with the following Redis commands:

```
> localhost connected!
> set name jack
OK
> get name
jack
> get abc
null
> setex code 30 1234
OK
```

Below the command history is a text input field with the placeholder text: '输入Redis命令后, 按Enter键执行, 上下键切换历史'.

The screenshot shows the Redis desktop manager interface. On the left, there's a sidebar with tabs for 'localhost' and 'DB0'. A search bar at the top says 'Enter 键进行搜索'. Below it, there are two entries: 'code' and 'name'. A button '加载更多' is visible. The main area shows a table with columns 'String' and 'code'. The 'code' row has a TTL field set to '-2'. A red box highlights this TTL field. To the right of the TTL field is a button labeled '距离失效时间'. At the bottom of the main area is a blue '保存' (Save) button.

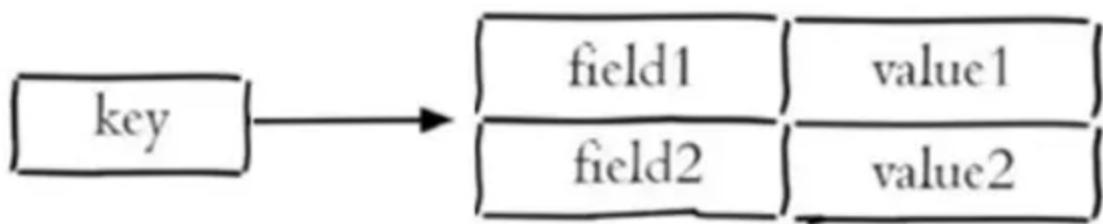
```
> setnx key1 itcast
1
> setnx key1 itheima
0
> get key1
itcast
```

哈希操作命令

Redis hash 是一个string类型的 field 和 value 的映射表，hash特别适合用于存储对象，常用命令：

- HSET key field value 将哈希表key中的字段field的值设为value
- HGET key field 获取存储在哈希表中指定字段
- HDEL key field 删除存储在哈希表中的指定字段
- HKEYS key 获取哈希表中所有字段
- HVALS key 获取哈希表中所有值

value



Screenshot of a Redis web interface. The top navigation bar shows "localhost" and "100 | localhost | DBO". The main area displays a table with two rows:

ID (Total: 2)	Key	Value	操作
1	name	xiaoming	编辑 面板 </>
2	age	22	编辑 面板 </>

The left sidebar shows the key "100" expanded, revealing sub-fields "code" and "name". A "加载更多" button is visible at the bottom of the sidebar.

```

> localhost connected!
> set name jack
OK
> get name
jack
> get abc
null
> setex code 30 1234
OK
> hset 100 name xiaoming
ERR unknown command 'hest'
> hset 100 name xiaoming
1
> hset 100 age 22
1
> hget 100
ERR wrong number of arguments for 'hget' command
> hget 100 name
xiaoming
> hkeys 100
name
age

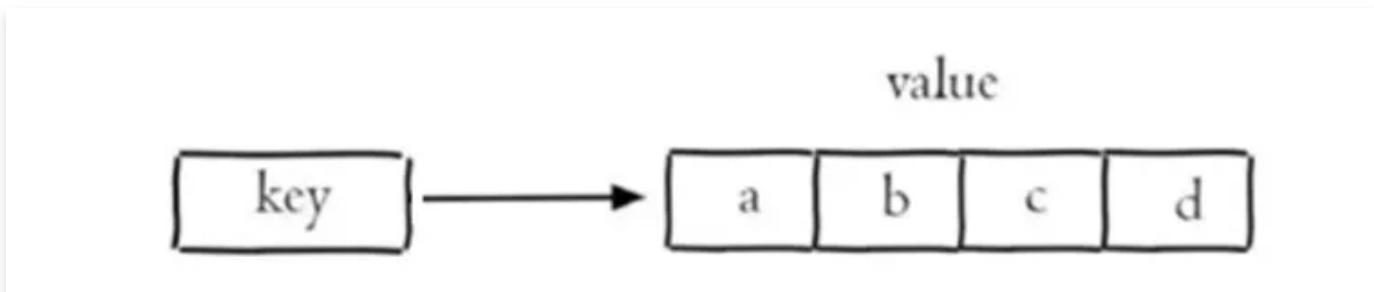
```

输入Redis命令后，按Enter键执行，上下键切换历史

列表操作命令

Redis 列表是简单的字符串列表，按照插入顺序排列，常用命令：

- LPUSH key value1 [value2] 将一个或多个值插入到列表头部
- LRANGE key start stop 获取列表指定范围内的元素
- RPOP key 移除并获取列表最后一个元素
- LLEN key 获取列表的长度



The RedisLab interface shows a connection to localhost with a selected database of DB0. A list named 'mylist' is displayed with the following data:

ID (Total: 3)	Value	Operation
1	d	
2	c	
3	b	

A "加载更多" (Load more) button is visible at the bottom.

The RedisLab interface shows a connection to localhost with a selected database of DB0. A terminal window displays the following Redis command history:

```

OK
> hest 100 name xiaoming
ERR unknown command 'hest'
> hset 100 name xiaoming
1
> hset 100 age 22
1
> hget 100
ERR wrong number of arguments for 'hget' command
> hget 100 name
xiaoming
> hkeys 100
name
age
> lpush mylist a b c
3
> lpush mylist d
4
> lrange mylist 0 -1
d
c
b
a
> rpop mylist
a
> llen mylist
3

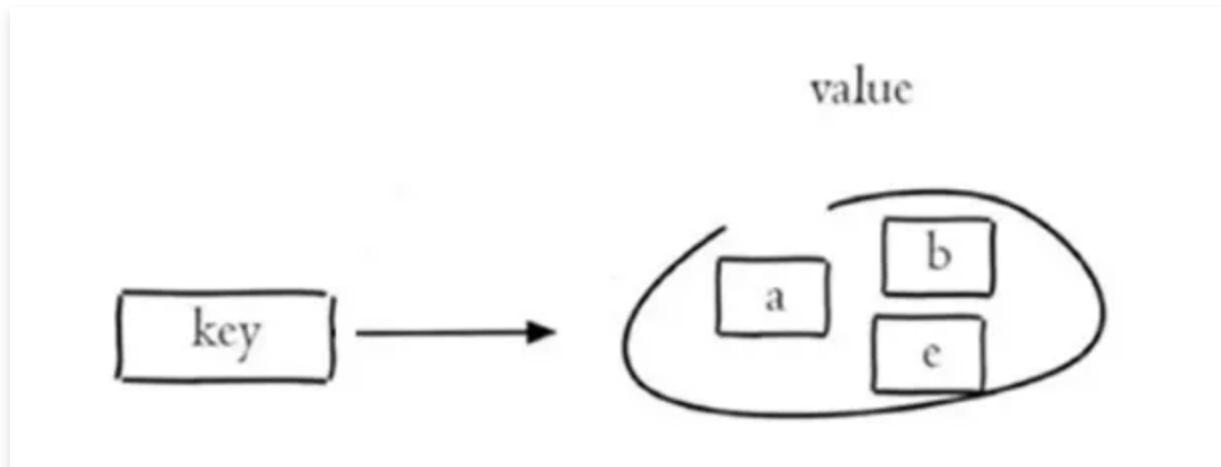
```

A text input field at the bottom is labeled "输入Redis命令后, 按Enter键执行, 上下键切换历史" (Input Redis command, press Enter to execute, use up/down keys to switch history).

集合操作命令

Redis set 是string 类型的无序集合，集合成员是唯一的，集合中不能出现重复的数据，常用的命令：

- SADD key member1 [member2] 向集合添加一个或多个成员
- SMEMBERS key 返回集合中的所有成员
- SCARD key 获取集合的成员数
- SINTER key1 [key2] 返回给定所有集合的交集
- SUNION key1 [key2] 返回所有给定集合的并集
- SREM key member1 [member2] 删除集合中一个或多个成员



The screenshot shows the Redis desktop manager interface. At the top, there are tabs for "localhost" (selected), "localhost" (unselected), "100 | localhost | DB0", and "mylist | localhost | DB0". On the left, there's a sidebar with a "新建连接" button and a search bar. Below the search bar is a dropdown menu set to "DB0" and a "+ 新增Key" button. The main area contains a list of keys: "100", "code", "mylist", "name", "set1", and "set2". A "加载更多" button is at the bottom of this list. On the right, the terminal window displays Redis commands and their results:

```
3
> sadd set1 a b c d
4
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd set1 a
0
> smembers set1
d
c
a
b
> scard set1
4
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd set2 a b x y
4
> sinter set1 set2
a
b
> sunion set1 set2
d
a
c
b
```

At the bottom of the terminal window, a tooltip says: "输入Redis命令后, 按Enter键执行, 上下键切换历史".

The screenshot shows a Redis desktop manager interface with a command history window. The title bar at the top has tabs for 'localhost' (selected), 'localhost' (with a red 'X'), '100 | localhost | DB0', and 'mylist | localhost | DB0'. The left sidebar lists keys: '100', 'code', 'mylist', 'name', 'set1', and 'set2'. The main area displays a command history:

```
> sadd set1 a
0
> smembers set1
d
c
a
b
> scard set1
4
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd set2 a b x y
4
> sinter set1 set2
a
b
> sunion set1 set2
d
a
c
b
y
x
> srem set1 a
1
```

At the bottom of the history window, there is a placeholder text: "输入Redis命令后, 按Enter键执行, 上下键切换历史".

The screenshot shows a Redis desktop manager interface with a sorted set key editor. The title bar at the top has tabs for 'localhost' (selected), 'localhost' (with a red 'X'), '100 | localhost | DB0', 'mylist | localhost | DB0', and 'set1 | localhost | DB0' (selected). The left sidebar lists keys: '100', 'code', 'mylist', 'name', 'set1' (selected), and 'set2'. The main area shows a table for the 'set1' key:

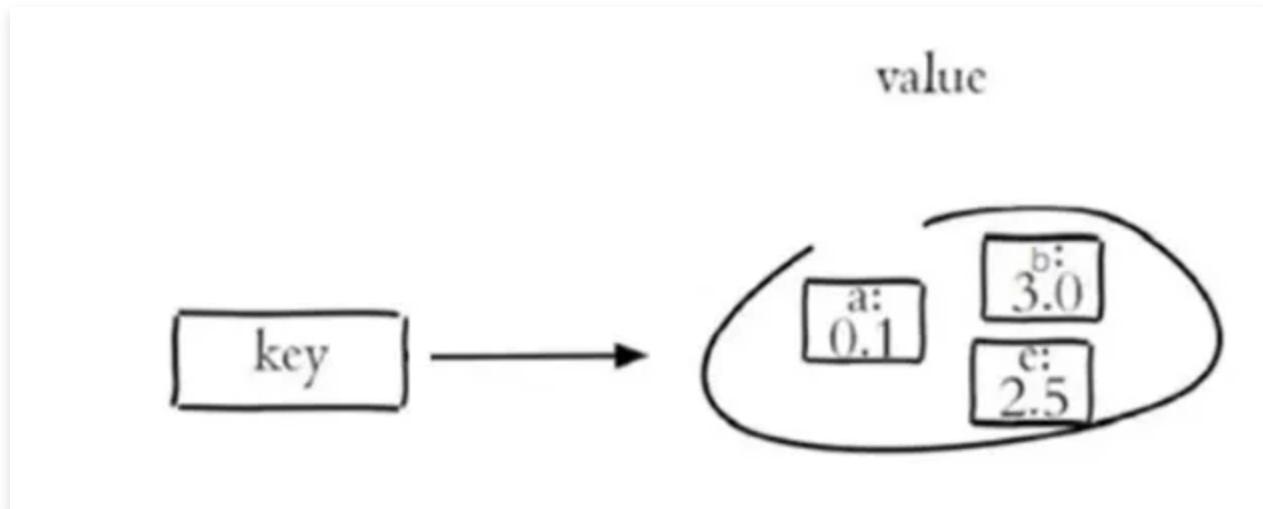
ID (Total: 3)	Value	操作
1	d	编辑 移除 <>
2	b	编辑 移除 <>
3	c	编辑 移除 <>

Below the table, there is a button labeled "添加新行" (Add New Row) and a "加载更多" (Load More) button.

有序集合操作命令

Redis有序集合是string类型元素的集合，且不允许有重复成员，每个元素都会关联一个double类型的分數常用，命令：

- ZADD key score1 member1 [score2 member2] 向有序集合添加一个或多个成员
- ZRANGE key start stop [WITHSCORES] 通过索引区间返回有序集合中指定区域内的成员
- ZINCRBY key increment member 有序集合中对指定成员的分数加上增量 increment
- ZREM key member [member ...] 移除有序集合中的一个或多个成员



The screenshot shows a Redis client interface with two main panes. The left pane displays a list of keys: 100, code, mylist, name, set1, set2, and zset1. Below this is a search bar with placeholder text 'Enter 键进行搜索' (Search key) and a '加载更多' (Load more) button. The right pane is a terminal window titled 'localhost' with the URL 'localhost'. It shows a command history and a scrollable text area. The history includes:

```
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd
ERR wrong number of arguments for 'sadd' command
> sadd set2 a b x y
4
> sinter set1 set2
a
b
> sunion set1 set2
d
a
c
b
y
x
> srem set1 a
1
> zadd
ERR wrong number of arguments for 'zadd' command
> zadd zset1 10.0 a 10.5 b
2
> zadd zset1 10.2 c
1
> zrange zset1 0 -1
a
c
b
> zrange zset1 0 -1 withscores
a
10
c
10.19999999999999
b
10.5
> zincrby zset1 5.0 a
15
```

At the bottom of the terminal window, there is a prompt: '输入Redis命令后, 按Enter键执行, 上下键切换历史' (Input Redis command, press Enter to execute, up and down keys to switch history).

The screenshot shows the Redis desktop manager interface. On the left, a sidebar lists keys: 100, code, mylist, name, set1, set2, and zset1, with zset1 selected. The main area displays a table for the 'zset1' key. The table has columns: ID (Total: 3), Score, Member, and a search bar. The data rows are:

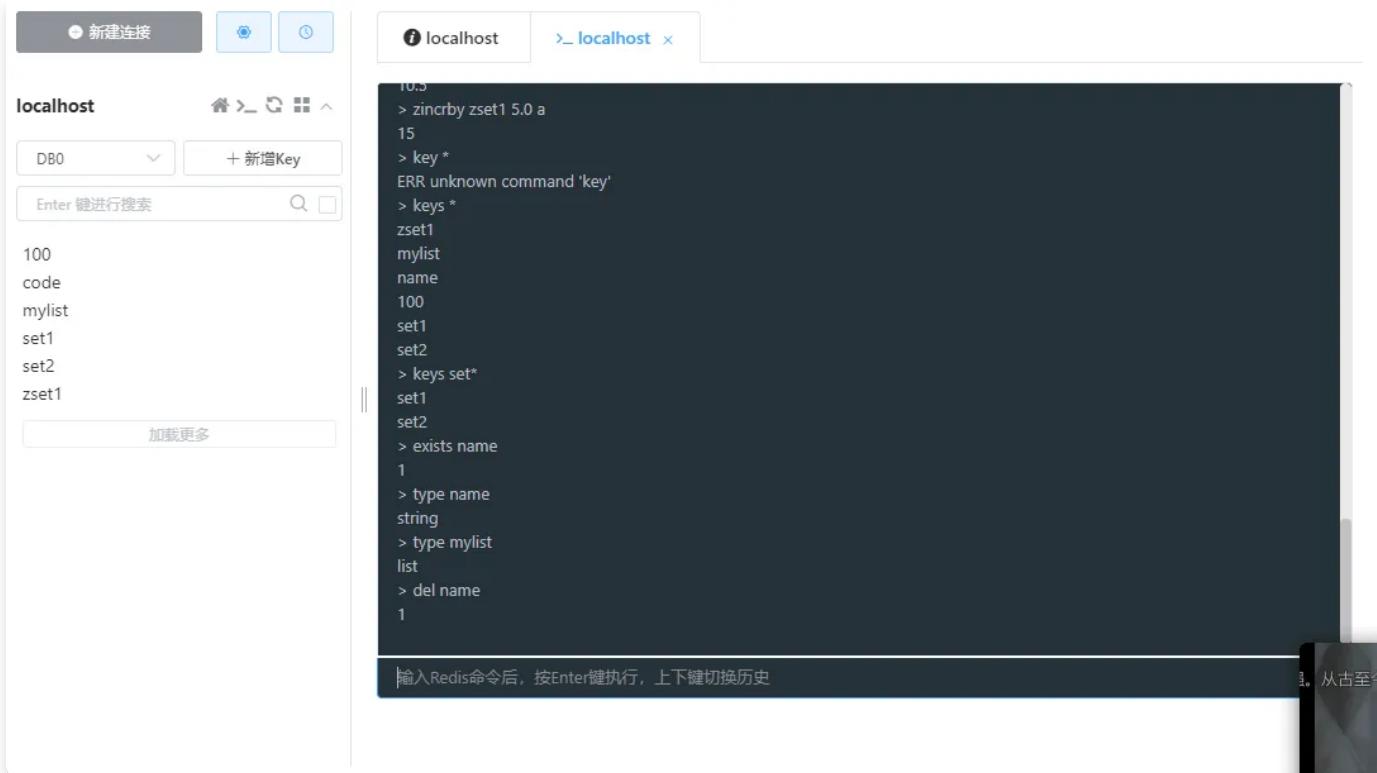
ID	Score	Member
1	15	a
2	10.5	b
3	10.2	c

At the bottom right of the table area is a '加载更多' (Load More) button.

通用命令

Redis 的通用命令是不分数据类型的，都可以使用的命令：

- **KEYS pattern** 查找所有符号给定模式 (pattern) 的key
- **EXISTS key** 检查给定key是否存在
- **TYPE key** 返回key所存储的值的类型
- **DEL key** 该命令用于在key存在时删除key



在Java中操作Redis

- Redis 的java客户端
- Spring Data Redis使用方式

Redis的java客户端

常见的几种java客户端

- Jedis
- Lettuce
- Spring Data Redis

Spring Data Redis是Spring 的一部分，对Redis底部开发包进行了高度封装。在Spring项目中，可以使用Spring Data Redis来简化操作

Spring Data Redis使用方式

操作步骤：

1. 导入Spring Data Redis 的maven坐标

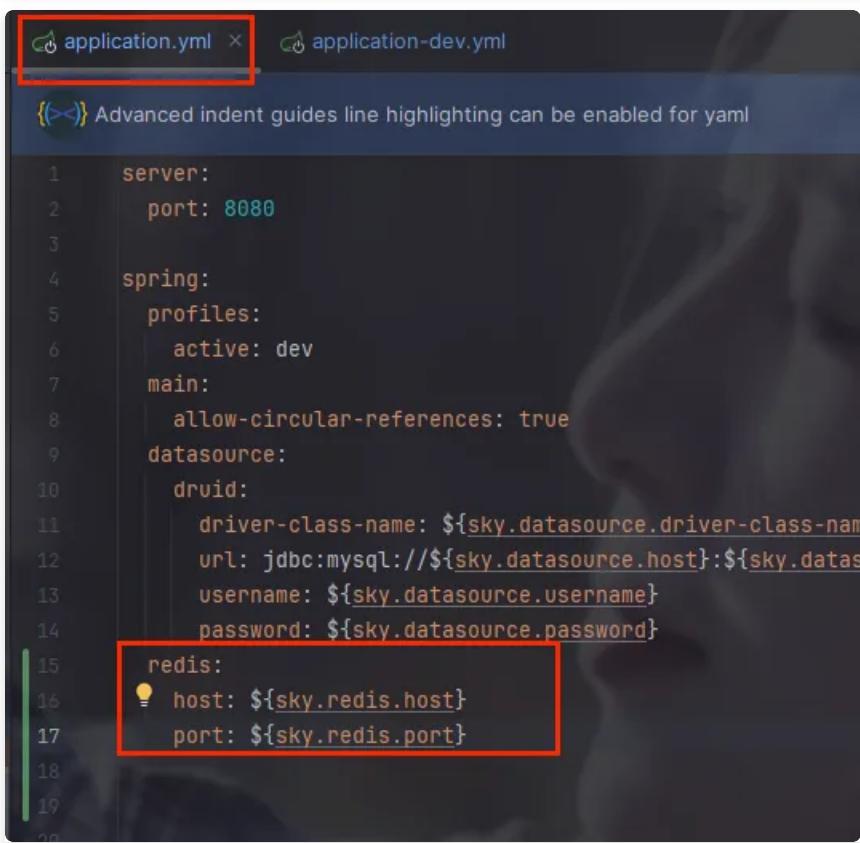
2. 配置Redis数据源
3. 编写配置类，创建RedisTemplate对象
4. 通过RedisTemplate对象操作Redis

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

pom.xml

```
spring:
  redis:
    host: localhost
    port: 6379
    password: 123456
```

application.yml



```
application.yml × application-dev.yml

{(><) Advanced indent guides line highlighting can be enabled for yaml

1   server:
2     port: 8080
3
4   spring:
5     profiles:
6       active: dev
7     main:
8       allow-circular-references: true
9     datasource:
10      druid:
11        driver-class-name: ${sky.datasource.driver-class-name}
12        url: jdbc:mysql://${sky.datasource.host}:${sky.datasource.port}
13        username: ${sky.datasource.username}
14        password: ${sky.datasource.password}
15      redis:
16        host: ${sky.redis.host}
17        port: ${sky.redis.port}
```

```

1 sky:
2   datasource:
3     driver-class-name: com.mysql.cj.jdbc.Driver
4     host: localhost
5     port: 3306
6     database: sky_take_out
7     username: root
8     password: 123456
9   aliasss:
10    endpoint: oss-cn-shenzhen.aliyuncs.com
11    access-key-id: LTAI5tJAHgH68J5gg8npsqFF
12    bucket-name: sky-itcast333333
13    access-key-secret: TmXptT3ZW6ZNWXKoLB1kdpmódbjqNW
14
15 redis:
16   host: localhost
17   port: 6379

```

```

import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.StringRedisSerializer;

@Configuration
public class RedisConfiguration {
    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory) {
        RedisTemplate<String, Object> redisTemplate = new RedisTemplate();
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        return redisTemplate;
    }
}

```

控制台输出：

```

2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] c.sky.mapper.DishMapper.pageQuery  : <==    Total: 1
2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] com.sky.mapper.DishMapper.pageQuery  : ===> Preparing: select d.* , c.name as categoryName from dish d left outer join category c on d.category_id = c.id order by d.create_time desc
2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] com.sky.mapper.DishMapper.pageQuery  : ===> Parameters: 10(Integer)
2024-08-21 13:53:04.184 DEBUG 20288 --- [nio-8080-exec-9] com.sky.mapper.DishMapper.pageQuery  : <==    Total: 10
当前线程数: 63
2024-08-21 14:30:41.944  WARN 20288 --- [nio-8080-exec-3] o.s.web.servlet.PageNotFound          : No mapping for GET /admin/setmeal/page
2024-08-21 14:30:41.944  INFO 20288 --- [nio-8080-exec-8] c.s.i.JwtTokenAdminInterceptor         : jwt校验:eyJhbGciOiJIUzI1NiJ9.eyJsbXBJZCIGMSwiZXhwIjoxNzI0MjI2NzYfQ.TUuZU-TvFfyRYiBnPqt-nr1H_7ee8M780IkVtAYZZCo
2024-08-21 14:30:41.944  INFO 20288 --- [nio-8080-exec-8] c.s.i.JwtTokenAdminInterceptor         : 当前员工Id:
2024-08-21 14:30:41.950 DEBUG 20288 --- [nio-8080-exec-8] com.sky.mapper.CategoryMapper.list    : ===> Preparing: select * from category where status = 1 and type = ? order by sort asc,create_time desc
2024-08-21 14:30:41.950 DEBUG 20288 --- [nio-8080-exec-8] com.sky.mapper.CategoryMapper.list    : ===> Parameters: 2(Integer)
2024-08-21 14:30:41.952 DEBUG 20288 --- [nio-8080-exec-8] com.sky.mapper.CategoryMapper.list    : <==    Total: 2

```

```
1 package com.sky.test;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.data.redis.core.*;
7
8 import java.util.concurrent.TimeUnit;
9
10 @SpringBootTest
11 public class SpringDataRedisTest {
12
13     @Autowired
14     private RedisTemplate redisTemplate;
15
16     @Test
17     public void testRedisTemplate() {
18         System.out.println(redisTemplate);
19         ValueOperations valueOperations = redisTemplate.opsForValue();
20         HashOperations hashOperations = redisTemplate.opsForHash();
21         ListOperations listOperations = redisTemplate.opsForList();
22         SetOperations setOperations = redisTemplate.opsForSet();
23         ZSetOperations zSetOperations = redisTemplate.opsForZSet();
24     }
25     /**
26      * 操作字符串类型的数据
27      */
28     @Test
29     public void testString() {
30         // set get setex setnx
31         redisTemplate.opsForValue().set("city", "北京");
32         String city = (String) redisTemplate.opsForValue().get("city");
33         System.out.println(city);
34         redisTemplate.opsForValue().set("code", "1234", 3, TimeUnit.MINUTES
35 );
36         redisTemplate.opsForValue().setIfAbsent("lock", "1");
37         redisTemplate.opsForValue().setIfAbsent("lock", "2");
38     }
39     /**
40      * 操作哈希类型的数据
41      */
42     @Test
43     public void testHash(){
44         //hset hget hdel hkeys hvals
```

```

45     HashOperations hashOperations = redisTemplate.opsForHash();
46
47     hashOperations.put("100","name","tom");
48     hashOperations.put("100","age","20");
49
50     String name = (String) hashOperations.get("100","name");
51     System.out.println(name);
52
53     Set keys = hashOperations.keys("100");
54     System.out.println(keys);
55
56     List values = hashOperations.values("100");
57     System.out.println(values);
58
59     hashOperations.delete("100","age");
60 }
61 /**
62 * 操作集合类型的数据
63 */
64 @Test
65 public void testSet(){
66     //sada smembers scard sinter sunion srem
67     SetOperations setOperations = redisTemplate.opsForSet();
68
69     setOperations.add("set1", "a","b","c","d");
70     setOperations.add("set2","a","b","x","y");
71
72     Set members = setOperations.members("set1");
73     System.out.println(members);
74
75     Long size = setOperations.size("set1");
76     System.out.println(size);
77
78     Set intersect = setOperations.intersect("set1","set2");
79     System.out.println(intersect);
80
81     Set union = setOperations.union("set1","set2");
82     System.out.println(union);
83
84     setOperations.remove("set1","a","b");
85 }
86
87 /**
88 *
89 */
90 @Test
91 public void testZSet(){
92     //zadd zrange zincrby zrem

```

```

93     ZSetOperations zSetOperations = redisTemplate.opsForZSet();
94
95     zSetOperations.add("zset1", "a", 10);
96     zSetOperations.add("zset1", "b", 12);
97     zSetOperations.add("zset1", "c", 9);
98
99     Set zset1 = zSetOperations.range("zset1", 0, -1);
100    System.out.println(zset1);
101
102    zSetOperations.incrementScore("zset1", "c", 10);
103
104    zSetOperations.remove("zset1", "a", "b");
105  }
106
107
108 /**
109  * 通用命令操作
110 */
111 @Test
112 public void testCommon(){
113     //keys exists type del
114     Set keys = redisTemplate.keys("*");
115     System.out.println(keys);
116
117     Boolean name = redisTemplate.hasKey("name");
118     Boolean age = redisTemplate.hasKey("set1");
119
120     for(Object key : keys){
121         DataType type = redisTemplate.type(key);
122         System.out.println(type.name());
123     }
124
125     redisTemplate.delete("mylist");
126
127 }
128 }
129

```

